Date	18 November 2022
Team ID	PNT2022TMID25137
Project Name	Car Resale Value Prediction

MODEL BUILDING:-

Choose The Appropriate Model:-

```
from sklearn.ensemble import ExtraTreesRegressor
model=ExtraTreesRegressor()
model.fit(X,y)


# In[64]:


print(model.feature_importances_)


# In[65]:


# plot graph of feature importance for visualzation
feat_importances=pd.Series(model.feature_importances_,index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()


# In[67]:


from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)


# In[69]:


X_train.shape


# In[70]:


from sklearn.ensemble import RandomForestRegressor
rf_random = RandomForestRegressor()
```

Check The Metrics Of The Model:-

```
# In[78]:


# Hyperparameters
# Randomized Search CV
# Number Of trees in  random forest
import numpy as np

n_estimators=[int(x) for x in np.linspace(start = 100,stop = 1200,num = 12)]

#Number of features to consider at every split
max_features=['auto','sqrt']

# Maximum number of levels in a tree
max_depth =[int(x) for x in np.linspace(5, 30,num =6)]

# max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split =[2,5,10,15,100]

# Minimum number of samples required to split each leaf node
min_samples_leaf = [1,2,5,10]


# In[79]:


from sklearn.model_selection import RandomizedSearchCV


# In[80]:


# create random grid
random_grid = {'n_estimators':n_estimators,
               'max_features':max_features,
               'max_depth':max_depth,
               'min_samples_split':min_samples_split,
               'min_samples_leaf':min_samples_leaf}

print(random_grid)


# In[83]:
```

Save The Model:-

```
# use random grid  to search for best hyperparameters
# first create the base model to tune
rf=RandomForestRegressor()


# In[85]:


rf_random = RandomizedSearchCV(estimator = rf,param_distributions =
random_grid,scoring
='neg_mean_squared_error',n_iter=10,cv=5,verbose=2,random_state=42,n_jobs =1)


# In[86]:


rf_random.fit(X_train,y_train)


# In[87]:


predictions = rf_random.predict(X_test)


# In[88]:


predictions


# In[89]:


sns.distplot(y_test-predictions)


# In[90]:


plt.scatter(y_test,predictions)


# In[92]:


import pickle
```

```
# open a file where you want to store data
file=open('random_forest_regression_model.pkl' , 'wb')

#dump information to that file
pickle.dump(rf_random,file)
```